Robust Analysis of Timed Automata *via* Channel Machines*

Patricia Bouyer[1,2,**], Nicolas Markey[1], Pierre-Alain Reynier[3,***]

[1] LSV, CNRS & ENS de Cachan, France
[2] Oxford University Computing Laboratory, UK
[3] Université Libre de Bruxelles, Belgium
{bouyer,markey}@lsv.ens-cachan.fr, reynier@ulb.ac.be

Abstract. Whereas formal verification of timed systems has become a very active field of research, the idealised mathematical semantics of timed automata cannot be faithfully implemented. Several works have thus focused on a modified semantics of timed automata which ensures implementability, and robust model-checking algorithms for safety, and later LTL properties have been designed. Recently, a new approach has been proposed, which reduces (standard) model-checking of timed automata to other verification problems on channel machines. Thanks to a new encoding of the modified semantics as a network of timed systems, we propose an original combination of both approaches, and prove that robust model-checking for coFlat-MTL, a large fragment of MTL, is EXPSPACE-Complete.

1 Introduction

Verification of real-time systems. In the last thirty years, formal verification of reactive systems has become a very active field of research in computer science. It aims at checking that (the model of) a system satisfies (a formula expressing) its specifications. The importance of taking real-time constraints into account in verification has quickly been understood, and the model of timed automata [2] has become one of the most established models for real-time systems, with a well-studied underlying theory, the development of mature model-checking tools (Uppaal [21], Kronos [11], ...), and numerous success stories.

Implementation of real-time systems. Implementing mathematical models on physical machines is an important step for applying theoretical results on practical examples. This step is well-understood for many untimed models that have been studied (*e.g.*, finite automata, pushdown automata). In the timed setting, while timed automata are widely-accepted as a framework for modelling real-time aspects of systems, it is known that they cannot be faithfully implemented

* Partly supported by project DOTS (ANR-06-SETIN-003).
** Partly supported by a Marie Curie fellowship (European Commission).
*** Partly supported by a Lavoisier fellowship (French Ministry of Foreign Affairs).

on finite-speed CPUs [12]. Studying the "implementability" of timed automata is thus a challenging question of obvious theoretical and practical interest.

A semantical approach. In [16], a new semantics for timed automata is defined that takes into account the digital aspects of the hardware on which the automaton is being executed. A timed automaton is then said to be *implementable* if, under this new semantics, the behaviours of this automaton satisfies its specifications. In order to study it efficiently, this semantics is over-approximated by the *AASAP* (for *Almost ASAP*), which consists in "enlarging" the constraints on the clocks by some parameter δ. For instance, "$x \in [a, b]$" is transformed into "$x \in [a-\delta, b+\delta]$". Implementability can be ensured by establishing the existence of some positive δ for which the AASAP semantics meets the specification. The decidability of this "*robust model-checking*" problem for specifications given as LTL formula has already been solved (first basic safety properties in [14] and then full LTL [9]) using a graph algorithm on the region automaton abstraction.

Timed temporal logics. Until recently [23], linear-time timed temporal logics were mostly considered as undecidable, and only MITL, the fragment without punctuality of MTL [20], was recognised as really tractable and useful [3]. Very recently [7], another fragment of MTL, called coFlat-MTL, has been defined, whose model-checking is EXPSPACE-Complete. The decidability of this logic relies on a completely original method using channel machines.

Our contribution. Inspired by the channel machine approach of [7], we propose a new techniques to robust model-checking of linear-time timed temporal logics. It is based on the construction of a network of timed systems which captures the AASAP semantics, and which can be expressed as a channel machine. Based on this approach, we prove that the robust model-checking of coFlat-MTL is EXPSPACE-Complete, *i.e.*, not more expensive than under the classical semantics. It is worth noticing that coFlat-MTL includes LTL, our result thus encompasses the previously shown decidability results in that framework.

Related work. Since its definition in [16], the approach based on the AASAP semantics has received much attention, and even other kind of perturbations like the drift of clocks, have been studied [26, 15, 4, 17]. In the case of safety properties and under some natural assumptions, this approach is equivalent to constraint enlargement and relies on similar techniques, as proved in [15]. Also, several works have proposed a symbolic, zone-based approach to the classical region-based algorithm for robustness [13, 27, 17]. This approach using the AASAP semantics contrasts with a modelling-based solution proposed in [1], where the behaviour of the platform is modelled as a timed automaton. Last, many other notions of "robustness" have been proposed in the literature in order to relax the mathematical idealisation of the semantics of timed automata [19, 22, 6, 5]. Those approaches are different from ours, since they roughly consist in dropping "isolated" or "unlikely" executions. Also note that robustness issues have also been handled in the untimed case, but are even further from our approach [18].

Outline of the paper. We introduce the setting in Section 2. Section 3 contains our construction: we first turn the robust semantics of timed automata into networks of timed systems (Section 3.1), which are then encoded as channel automata (Section 3.2). We then explain how the resulting channel automata are used for Bounded-MTL and coFlat-MTL model-checking (Section 3.3). By lack of space, proofs are omitted and can be found in the research report [10].

2 Preliminaries

We present here the model of timed automata, some linear-time timed temporal logics, and the model of channel automata, which is central to our approach.

2.1 Timed Automata

Let X be a finite set of *clock* variables. We denote by $\mathcal{G}(X)$ the set of *clock constraints* generated by the grammar $g ::= g \wedge g \mid\ x \sim k$, where $x \in X$, $k \in \mathbb{N}$, and $\sim \in \{\leq, \geq\}$. A *(clock) valuation* v for X is an element of $\mathbb{R}_+^X$. If $v \in \mathbb{R}_+^X$ and $t \in \mathbb{R}_+$, we write $v + t$ for the valuation assigning $v(x) + t$ to every clock $x \in X$. If $r \subseteq X$, $v[r \leftarrow 0]$ denotes the valuation assigning 0 to every clock in r and $v(x)$ to every clock in $X \setminus r$.

A *timed automaton* is a tuple $\mathcal{A} = (L, \ell_0, X, I, \Sigma, T)$ where L is a finite set of locations, $\ell_0 \in L$ is an initial location, X is a finite set of clocks, $I\colon L \to \mathcal{G}(X)$ labels each location with its invariant, Σ is a finite set of actions, and $T \subseteq L \times \mathcal{G}(X) \times \Sigma \times 2^X \times L$ is the set of transitions. Given a parameter value $\delta \in \mathbb{R}_{\geq 0}$, whether a valuation $v \in \mathbb{R}_+^X$ satisfies a constraint g within δ, written $v \models_\delta g$, is defined inductively as follows:

$$\begin{cases} v \models_\delta x \leq k & \text{iff } \ v(x) \leq k + \delta \\ v \models_\delta x \geq k & \text{iff } \ v(x) \geq k - \delta \\ v \models_\delta g_1 \wedge g_2 & \text{iff } \ v \models_\delta g_1 \text{ and } v \models_\delta g_2 \end{cases}$$

Following [16], we define a parameterised semantics for $\mathcal{A}$ as a timed transition system $[\![\mathcal{A}]\!]_\delta = \langle S, S_0, \Sigma, \to_\delta \rangle$. The set S of states of $[\![\mathcal{A}]\!]_\delta$ is $\{(\ell, v) \in L \times \mathbb{R}_+^X \mid v \models_\delta I(\ell)\}$, with $S_0 = \{(\ell_0, v_0) \mid v_0(x) = 0 \text{ for all } x \in X\}$. A transition in $[\![\mathcal{A}]\!]_\delta$ is composed either of a delay move $(\ell, v) \xrightarrow{d}_\delta (\ell, v + d)$, with $d \in \mathbb{R}_+$, when both v and $v{+}d$ satisfy the invariant $I(\ell)$ within δ, or of a discrete move $(\ell, v) \xrightarrow{\sigma}_\delta (\ell', v')$ when there exists a transition $(\ell, g, \sigma, r, \ell') \in T$ with $v \models_\delta g$, $v' = v[r \leftarrow 0]$, and $v' \models_\delta I(\ell')$. The graph $[\![\mathcal{A}]\!]_\delta$ is thus an infinite transition system. Notice that, in the definitions above, the standard semantics of timed automata can be recovered by letting $\delta = 0$. In that case, we omit the subscript δ.

A *run* of $[\![\mathcal{A}]\!]_\delta$ is an infinite sequence $(\ell_0, v_0) \xrightarrow{d_0}_\delta (\ell_0, v_0{+}d_0) \xrightarrow{\sigma_0}_\delta (\ell_1, v_1) \xrightarrow{d_1}_\delta (\ell_1, v_1 + d_1) \dots$ where for each $i \geq 0$, $d_i \in \mathbb{R}_+$. A *timed word* w is an infinite sequence $(\sigma_i, t_i)_{i \in \mathbb{N}}$ where $\sigma_i \in \Sigma$ and $t_i \in \mathbb{R}_+$ for each $i \geq 0$, and such that the sequence $(t_i)_{i \in \mathbb{N}}$ is non-decreasing and diverges to infinity. The timed word w is read on the run $(\ell_0, v_0) \xrightarrow{d_0}_\delta (\ell_0, v_0 + d_0) \xrightarrow{\sigma_0}_\delta (\ell_1, v_1) \xrightarrow{d_1}_\delta (\ell_1, v_1 + d_1) \dots$ whenever $t_i = \sum_{j \leq i} d_j$ for every $i \in \mathbb{N}$. We write $\mathcal{L}([\![\mathcal{A}]\!]_\delta)$ for the set of timed words that can be read on a run of $[\![\mathcal{A}]\!]_\delta$ starting in (ℓ_0, v_0). More generally, we write $\mathcal{L}([\![\mathcal{A}]\!]_\delta^{(\ell,v)})$ for the set of timed words than can be read starting from (ℓ, v).

Since our results rely on the results of [14, 9], we require that our timed automata satisfy the following requirements: *(i)* constraints in guards and invariants only involve non-strict inequalities; *(ii)* all the clocks are always bounded by some constant M; *(iii)* all the cycles in the region graph are *progress cycles*, *i.e.*, all the clocks are reset along those cycles. In addition, we require that the timed automata are *non-blocking*, in the sense that from every state, an action transition will eventually become firable: for every $(\ell, v) \in L \times \mathbb{R}_+^X$ such that $v \models_0 I(\ell)$, there exists $d \in \mathbb{R}_+^X$ and $\sigma \in \Sigma$ such that $(\ell, v) \xrightarrow{d}_0 (\ell, v{+}d) \xrightarrow{\sigma}_0 (\ell', v')$ for some $(\ell', v') \in L \times \mathbb{R}_+^X$.

2.2 Implementability and Robustness of Timed Automata

The parameterised semantics defined above (referred to as "enlarged semantics" in the sequel), has been defined in [16] in order to study the *implementability* of timed systems. Indeed, timed automata are governed by a mathematical, idealised semantics, which does not fit with the digital, imprecise nature of the hardware on which they will possibly be implemented. An *implementation semantics* has thus been defined in [16] in order to take the hardware into account: that semantics models a digital CPU which, every δ_P time units (at most), reads the value of the digital clock (updated every δ_L time units), computes the values of the guards, and fires one of the available transitions. We write $[\![\mathcal{A}]\!]^{\delta_P,\delta_L}$ for the resulting transition system, and $\mathcal{L}([\![\mathcal{A}]\!]^{\delta_P,\delta_L})$ for the corresponding set of timed words. Given a (linear-time) property $\mathcal{P}$, *i.e.*, a set of accepted timed words, a timed automaton $\mathcal{A}$ is said to be implementable w.r.t. $\mathcal{P}$ iff $\mathcal{L}([\![\mathcal{A}]\!]^{\delta_P,\delta_L}) \subseteq \mathcal{P}$ for some positive values of δ_P and δ_L.

As proved in [16], the enlarged semantics simulates the implementation semantics as soon as $\delta > 4\delta_P + 3\delta_L$. As a consequence, it is sufficient to check the existence of $\delta > 0$ such that $\mathcal{L}([\![\mathcal{A}]\!]_\delta) \subseteq \mathcal{P}$ in order to ensure implementability of $\mathcal{A}$ w.r.t. $\mathcal{P}$. We follow this idea in the sequel, and study the *robust satisfaction* relation: a timed automaton *robustly satisfies* a linear-time property $\mathcal{P}$, written $\mathcal{A} \models\!\!\!\equiv \mathcal{P}$, whenever $\mathcal{L}([\![\mathcal{A}]\!]_\delta) \subseteq \mathcal{P}$ for some $\delta > 0$.

2.3 Some Subclasses of Metric Temporal Logic

Linear-time properties are often defined *via* temporal logic formulae. In this paper, we focus on subclasses of MTL (Metric Temporal Logic) [20].

Fix a finite, non-empty alphabet Σ. The syntax of MTL over Σ is defined by the following grammar:

$$\mathsf{MTL} \ni \varphi ::= \sigma \mid \neg\sigma \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi\, \mathbf{U}_I\, \varphi \mid \varphi\, \widetilde{\mathbf{U}}_I\, \varphi$$

where $\sigma \in \Sigma$, and I is an interval of $\mathbb{R}^+$ with bounds in $\mathbb{N} \cup \{\infty\}$. MTL formulae are interpreted over timed words. Let $w = (\sigma_i, t_i)_{i \geq 0}$ be a timed word, and $p \in \mathbb{N}$.

The (pointwise) semantics of MTL is defined recursively as follows (we omit Boolean operations):

$$\begin{aligned} w,p \models \sigma &\Leftrightarrow \sigma_p = \sigma \\ w,p \models \varphi \,\mathbf{U}_I\, \psi &\Leftrightarrow \exists i > 0 \text{ s.t. } w,p+i \models \psi,\ t_{p+i} - t_p \in I \\ &\qquad \text{and } \ \forall 0 < j < i,\ w,p+j \models \varphi \\ w,p \models \varphi \,\widetilde{\mathbf{U}}_I\, \psi &\Leftrightarrow w,p \models \neg\Big((\neg\varphi)\,\mathbf{U}_I\,(\neg\psi)\Big). \end{aligned}$$

If $w,0 \models \varphi$, we write $w \models \varphi$. Following the discussion above, we write $\mathcal{A} \models\!\equiv \varphi$ if, for some $\delta > 0$, we have $w \models \varphi$ for every $w \in \mathcal{L}(\llbracket \mathcal{A} \rrbracket_\delta)$.

Additional operators, such as $\mathtt{tt}$ (true), $\mathtt{ff}$ (false), $\Rightarrow$, $\Leftrightarrow$, $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{X}$, are defined in the usual way: $\mathbf{F}_I\,\varphi \equiv \mathtt{tt}\,\mathbf{U}_I\,\varphi$, $\mathbf{G}_I\,\varphi \equiv \mathtt{ff}\,\widetilde{\mathbf{U}}_I\,\varphi$, and $\mathbf{X}_I\,\varphi \equiv \mathtt{ff}\,\mathbf{U}_I\,\varphi$. We also use pseudo-arithmetic expressions to denote intervals. For example, '$= 1$' denotes the singleton $\{1\}$.

Following [7], we identify the following syntactic fragments of MTL: LTL [25] can be considered as the fragment of MTL in which modalities are not constrained (*i.e.*, where $\mathbb{R}_+$ is the only constraining interval); $\mathsf{Bounded\text{-}MTL}$ is the fragment of MTL in which all interval constraints are bounded: observe that $\mathsf{Bounded\text{-}MTL}$ disallows unconstrained modalities, and is in particular not suitable to express invariance properties (the most basic type of temporal specifications). The fragment $\mathsf{coFlat\text{-}MTL}$[4] has then been defined to remedy this deficiency:

$$\mathsf{coFlat\text{-}MTL} \ni \varphi ::= \sigma \mid \neg\sigma \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi\,\mathbf{U}_J\,\varphi \mid \varphi\,\mathbf{U}_I\,\underline{\psi} \mid \varphi\,\widetilde{\mathbf{U}}_J\,\varphi \mid \underline{\psi}\,\widetilde{\mathbf{U}}_I\,\varphi$$

where J ranges over the set of bounded intervals, I over the set of all intervals, and the underlined formula $\underline{\psi}$ ranges over LTL.

One immediately sees that $\mathsf{coFlat\text{-}MTL}$ subsumes both LTL and $\mathsf{Bounded\text{-}MTL}$, but it is not closed under negation. However, it is closed under invariance, and can then express *e.g.* bounded response-time or even richer formulae such as $\mathbf{G}\,(\mathit{request} \Rightarrow \mathbf{F}_{\leq 5}\,(\mathit{acquire} \wedge \mathbf{F}_{=1}\,\mathit{release}))$.

2.4 Channel Automata

Channel automata are an interesting formalism for reasoning about alternating timed automata, which has been used in [7] to prove the $\mathsf{EXPSPACE}$-membership of the model-checking problem for $\mathsf{coFlat\text{-}MTL}$. We only give the definition and necessary results here, and refer to [10] for some more intuition.

A *channel automaton with renaming and occurrence testing* (CAROT for short) is a tuple $\mathcal{C} = (S, s_0, \Sigma, \delta, F)$, where S is a finite set of control states, $s_0 \in S$ is the initial control state, $F \subseteq S$ is a set of accepting control states, Σ is a finite alphabet, $\delta \subseteq S \times Op \times S$ is the set of transition rules, where

$$Op = \{\sigma!, \sigma? \mid \sigma \in \Sigma\} \cup \{\mathit{zero?}(\sigma) \mid \sigma \in \Sigma\} \cup \{R \mid R \subseteq \Sigma \times \Sigma\}$$

[4] We do not explain this terminology here, and refer to [7] for deeper considerations.

is the set of operations. Given a rule $\tau = (s, \alpha, s') \in \delta$, we define $op(\tau) = \alpha$.

Intuitively, operations $\sigma!$ and $\sigma?$ are the classical write and read operations, $zero?(\sigma)$ is a guard for testing the occurrence of σ in the channel, and $R \subseteq \Sigma \times \Sigma$ is interpreted as a global renaming. An *end-of-channel* marker can be used to count the number of times the whole channel is read: it suffices to add, from each state of the CAROT, an outgoing transition reading $\triangleright$, immediately followed by a transition writing $\triangleright$. That way, there is always a unique copy of $\triangleright$ on the channel (except when it has just been read). The number of transitions writing $\triangleright$ along a computation ϱ is the number of *cycles* of ϱ, denoted by $cycles(\varrho)$. In the sequel, the CAROTs are assumed to contain the end-of-channel marker $\triangleright$, and to have all their states equipped with a loop reading and writing that symbol.

We consider in the sequel a restricted version of the reachability problem for CAROTs, where we impose a bound on the "time" (measured here as the number of cycles of the channel): the *cycle-bounded reachability problem* for CAROTs is defined as follows: given a CAROT $\mathcal{C}$, an input word $w \in \Sigma^*$, and a cycle bound h, does $\mathcal{C}$ have an accepting computation ϱ on w with $cycles(\varrho) \leqslant h$?

In [7], a non-deterministic procedure is presented to check the existence of an accepting cycle-bounded computation using only polynomial space in the *value* of the cycle bound and in the size of the CAROT:

Theorem 1. *The cycle-bounded reachability problem for CAROTs is solvable in polynomial space in the size of the channel automaton, the size of the input word, and the value of the cycle bound.*

Remark. Note that the algorithm could easily be adapted to cope with the cycle-bounded reachability between two global states (*i.e.*, control state and content of the channel): it suffices to first set the initial content of the channel, and to add transitions that would, from any point, run one more cycle of the channel and store its whole content in the location.

3 Robust Model-Checking of coFlat-MTL

In this section, we prove the main results of this paper, namely that the robust model-checking problem can be expressed using CAROTs, and then that the robust model-checking problem of coFlat-MTL is EXPSPACE-Complete. EXPSPACE-Hardness is a consequence of the EXPSPACE-Hardness of the satisfiability problem for Bounded-MTL [7]. EXPSPACE-membership is more involved and will be done in several steps:

1. We first construct a family of networks of timed systems that captures the enlarged semantics of timed automata (Subsection 3.1).
2. We then transform this family of networks into a CAROT, that will simulate the joint behaviours of those networks of timed systems with an alternating timed automaton representing the property we want to robustly verify (Subsection 3.2).

3. Finally, we use the CAROT to design a decidability algorithm for the robust model-checking problem, first for Bounded-MTL, and then for coFlat-MTL (Subsection 3.3).

For the rest of the paper, we fix a timed automaton $\mathcal{A} = (L, \ell_0, X, I, \Sigma, T)$.

3.1 From Robustness to Networks of Timed Systems

In this subsection, we transform the robust model-checking problem into a model-checking problem in an infinite family of timed systems. The correctness of the transformation relies on simulation relations between timed transition systems: given two timed transition systems $\mathcal{T}_i = (S_i, \Sigma, \to_i)$ for $i = 1, 2$, we say that $\mathcal{T}_2$ *simulates* $\mathcal{T}_1$ whenever there exists a non-empty relation $\mathfrak{R} \subseteq S_1 \times S_2$ such that $(s_1, s_2) \in \mathfrak{R}$ and $s_1 \xrightarrow{\alpha}_1 s_1'$ with $\alpha \in \Sigma \cup \mathbb{R}_+$ implies $s_2 \xrightarrow{\alpha}_2 s_2'$ for some $s_2' \in S_2$ with $(s_1', s_2') \in \mathfrak{R}$. We then write $\mathcal{T}_1 \sqsubseteq \mathcal{T}_2$.

Let n be an integer; we denote by $\mathcal{B}^n$ the timed network composed of the following components (which are not standard timed automata because of the use of disjunction and fractional parts in the guards):

• for each $0 \leq i < n$, $\mathcal{B}_i$ is the timed automaton depicted on Fig. 1, where x_i is a fresh clock not belonging to X, and $[x_i \leq 1]$ is an invariant forbidding the clock x_i become larger than 1. That way, this automaton is forced to fire its transition when the value of x_i reaches 1. We call such an automaton a Δ-*automaton* in the sequel. In the following, we will take indices of clocks x_i modulo n, and in particular $x_{i+1} = x_0$ whenever $i = n - 1$.

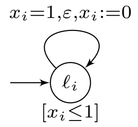

Fig. 1. Automaton $\mathcal{B}_i$

• the timed automaton $\mathcal{B}$, built from $\mathcal{A}$ by modifying the guards and invariants as follows: each constraint of the form $x \leq k$ is replaced with

$$(x < k+1) \ \wedge \ \Big(x > k \ \Rightarrow \bigvee_{0 \leq i < n} \{x\} \leq x_{i+1} < x_{i-1}\Big)$$

and each constraint of the form $x \geq k$ is replaced with

$$(x > k-1) \ \wedge \ \Big(x < k \ \Rightarrow \bigvee_{0 \leq i < n} \{x\} \geq x_{i-1} > x_{i+1}\Big)$$

We need to explain when a valuation v satisfies these "extended" guards. Boolean operators are handled in the natural way, and $\{x\}$ is intended to denote the fractional part of the value of clock x.

It naturally defines a timed transition system $[\![\mathcal{B}^n]\!]$, as the synchronised product (synchronised because of the time) of all components. Denoting by $X_n = X \cup \{x_i \mid 0 \leq i < n\}$ the set of clocks of $\mathcal{B}^n$, a configuration of $[\![\mathcal{B}^n]\!]$ can be described by a pair (ℓ, v) where $\ell \in L$ and $v \in \mathbb{R}_+^{X_n}$ (each Δ-automaton has a single location). We write u_n for the valuation over X_n assigning 0 to clocks in X and $\frac{i}{n}$ to every clock x_i for $0 \leq i < n$. The initial configuration of this timed network is the pair (ℓ_0, u_n). Delay and action transitions are defined naturally in the synchronised products of all the components. However in the following we will hide ε-moves due to the components $\mathcal{B}_i$. Thus, in $[\![\mathcal{B}^n]\!]$, we write $(\ell, v) \overset{t}{\Rightarrow} (\ell, v')$ for an interleaving of delay transitions (which sum up to t) and of ε-moves in the $\mathcal{B}_i$'s. For uniformity, we write $\overset{\sigma}{\Rightarrow}$ for σ-moves in $[\![\mathcal{B}^n]\!]$. In the sequel, simulation relations assume the relation $\Rightarrow$ in $[\![\mathcal{B}^n]\!]$, and the simple transition relation $\rightarrow_\delta$ in $[\![\mathcal{A}]\!]_\delta$. In the same way, the intended language accepted by $[\![\mathcal{B}^n]\!]$ should ignore ε-transitions. In other words, it should also be defined using the relation $\Rightarrow$ as the transition relation:

$$\mathcal{L}([\![\mathcal{B}^n]\!]) = \{w = (\sigma_i, t_i)_{i \in \mathbb{N}} \mid \exists (\ell_0, u_n) \overset{d_0}{\Longrightarrow} (\ell_0, u') \overset{\sigma_0}{\Longrightarrow} (\ell_1, u'') \overset{d_1}{\Longrightarrow} \cdots \in [\![\mathcal{B}^n]\!] \\ \text{s.t. } \forall i \in \mathbb{N},\ t_i = \sum_{j \leq i} d_j\}$$

Lemma 2. *For every $n \geq 3$, $[\![\mathcal{A}]\!]_{\frac{1}{n}} \sqsubseteq [\![\mathcal{B}^n]\!] \sqsubseteq [\![\mathcal{A}]\!]_{\frac{2}{n}}$.*

With the previous definition, the simulation results can be stated in terms of language inclusion (as initial states are preserved by the exhibited simulation relations): for every $n \geq 3$, $\mathcal{L}([\![\mathcal{A}]\!]_{\frac{1}{n}}) \subseteq \mathcal{L}([\![\mathcal{B}^n]\!]) \subseteq \mathcal{L}([\![\mathcal{A}]\!]_{\frac{2}{n}})$.

Theorem 3. *Let $\varphi \in \mathsf{MTL}$. Then, $\mathcal{A} \models\!\!\equiv \varphi \;\Leftrightarrow\; \exists n \geq 3$ s.t. $[\![\mathcal{B}^n]\!] \models \varphi$.*

3.2 From Networks of Timed Systems to CAROTs

Extending the approach of [7], the CAROT we build is such that it accepts joint executions of the network of timed systems we just built (and not of a single timed automaton as in [7]) and of the alternating timed automaton corresponding to the negation of the coFlat-MTL formula we want to verify. In order to handle arbitrarily many components in the network, and to deal with "extended" guards (*i.e.*, with disjunctions and fractional parts), the construction attached to the network of timed systems needs to be deeply modified. In a first step, we describe a CAROT that only encodes the behaviours of the network of timed systems.

Let M be the maximal constant appearing in the automaton $\mathcal{A}$. Then $M+1$ is larger than or equal to the maximal constant of any $\mathcal{B}^n$. We assume that the clocks of $\mathcal{A}$ (and $\mathcal{B}^n$) take their values in $[0, M+1] \cup \{\perp\}$, where $\perp$ is a special value (intended to represent any value larger than $M+1$). We write Reg for the set $\{0, 1, \ldots, M+1, \perp\}$ and $\Lambda = \wp(L \times X \times \mathsf{Reg})$.

A configuration $(\ell, v) \in L \times \mathbb{R}_+^{X_n}$ of the network of timed systems is encoded as the element $C_{(\ell,v)} = \{(\ell, x, v(x)) \mid x \in X_n\}$, and partitioned into a sequence of disjoint subsets C_0, C_1, ..., C_p, $C_\perp$, obtained using standard region techniques. More precisely, C_0 (resp. $C_\perp$) contains elements whose fractional part is 0 (resp. whose value is $\perp$) and the others C_i gather elements with the same fractional part and are sorted according to the increasing order of fractional parts. We then let $H(C_{(\ell,v)}) = \big(\mathsf{reg}(C_0),\ \mathsf{reg}(C_\perp),\ \mathsf{reg}(C_1)\mathsf{reg}(C_2)\ldots\mathsf{reg}(C_p)\big) \in \Lambda \times \Lambda \times \Lambda^*$, where we write $\mathsf{reg}(C)$ for $\{(\ell, x, \mathsf{reg}(v)) \mid (\ell, x, v) \in C\}$, with $\mathsf{reg}(v)$ the largest element of Reg smaller than or equal to v.

Using the abstraction function H, it is possible to define a discrete transition system $\mathcal{T}_H^n$ which abstracts away precise timing information, but which simulates the behaviours of $\mathcal{B}^n$. The abstraction function also provides an equivalence relation $\equiv$ on configurations: $C \equiv C'$ iff $H(C) = H(C')$. Extending straightforwardly a result of [24] (regions are compatible with our extended guards), we have:

Lemma 4. *The equivalence relation $\equiv$ is a time-abstract*[5] *bisimulation.*

The CAROT we will build is based on the above discrete transition system. More precisely, the intended encoding of a configuration in the CAROT is the following: the integral values of the clocks of $\mathcal{A}$ are stored in the discrete state of the CAROT, as well as the sets C_0 and $C_\perp$, and the current location ℓ. The other C_i's are stored on the channel, from C_1 (recently written on the channel) to C_p (the next item to be read from the channel).

In order to use the same CAROT to simulate the network of timed system with any number of Δ-automata, we abstract the name of clocks x_i in our encoding, representing them on the channel by a new symbol Δ. Since, at any time, at most one of the x_i's can have integral value, and since we only need to know the order of the x_i's in order to evaluate guards of $\mathcal{B}^n$, the amount of information to be stored in the location of the CAROT does not depend on the number of Δ-automata in the network.

Example 5. Consider for instance a configuration C encoded by the word

$$H(C) = \big(\{(\ell, x, 3), (\ell, x_2, 0)\},\ \{(\ell, z, \perp)\},\ \{(\ell, x_0, 0)\} \cdot \{(\ell, y, 1), (\ell, x_1, 0)\}\big).$$

We assume that the maximal constant M is 3. The encoding of the delay-successor of C is obtained by cycling around the letters (except the last one) of the word (and increasing the values of the regions accordingly). Writing $\varnothing$ for the empty set, the first delay successor of $H(C)$ is

$$\big(\varnothing,\ \{(\ell, z, \perp)\},\ \{(\ell, x, 3), (\ell, x_2, 0)\} \cdot \{(\ell, x_0, 0)\} \cdot \{(\ell, y, 1), (\ell, x_1, 0)\}\big).$$

The next delay successor is

$$\big(\{(\ell, y, 2), (\ell, x_1, 0)\},\ \{(\ell, z, \perp)\},\ \{(\ell, x, 3), (\ell, x_2, 0)\} \cdot \{(\ell, x_0, 0)\}\big).$$

Note that, in that second delay transition, we have reset clock x_1 when it has reached 1, and the integral part of y has increased to 2. The configuration $H(C)$ would be encoded as depicted on Fig. 2 (where we write to the left and read from the right of the channel). With respect to the channel, the first delay transition is performed through the write operation '$\langle x\Delta\rangle!$'. As in [7], it is worth noticing that a cycle of the CAROT corresponds to one time unit elapsing.

[5] This means that precise delays of time-elapsing transitions are abstracted away.

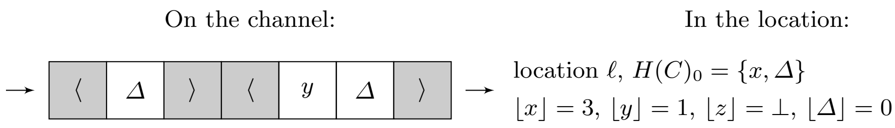

Fig. 2. Encoding of a configuration in a CAROT

Furthermore, to simulate the extended guards used in $\mathcal{B}^n$, we need some additional information about the position of clocks of $\mathcal{A}$ w.r.t. symbols Δ. As we have already seen, a clock x verifies a constraint $\{x\} \leq x_{i+1} < x_{i-1}$ iff its fractional part is smaller than one of the two smallest clocks x_j. In our simulation, this corresponds as being "before" the second symbol Δ on the channel. And symmetrically for constraint $\{x\} \geq x_{i-1} > x_{i+1}$. We thus have to store in the control part of the CAROT which clocks are "before" (resp. "after") the two first (resp. last) symbols Δ. Whereas this can easily be done for the clocks that have been recently written on the channel, this is not possible for the clocks lying at the head of the channel (this would require to store the position of each clock w.r.t. each symbol Δ). Instead, we use non-determinism to allow the CAROT make predictions about the content of the head of the channel, and then we verify when reading clocks from the channel that these predictions were correct.

For lack of space, we cannot present the formal construction of the CAROT, but report to [10]. We write $\mathcal{C}_{\mathcal{A}}$ for the resulting CAROT, $\mathcal{T}^n_{\mathcal{C}_{\mathcal{A}}}$ for the transition system associated with $\mathcal{C}_{\mathcal{A}}$ and restricted to configurations with correct predictions and n occurrences of Δ on the channel (or in the location), and $\approx$ for the relation that describes which configuration (d, c) of $\mathcal{T}^n_{\mathcal{C}_{\mathcal{A}}}$ (a control state together with a channel content) corresponds to a configuration of $\mathcal{T}^n_H$. The correctness of the construction relies on the following lemmas.

Lemma 6. *For any $n \geq 3$, the transition systems $\mathcal{T}^n_H$ and $\mathcal{T}^n_{\mathcal{C}_{\mathcal{A}}}$ are bisimilar.*

Lemma 7. *Let ϱ be a time-divergent execution in $[\![\mathcal{B}^n]\!]$. Then any computation in $\mathcal{C}_{\mathcal{A}}$ simulating ϱ has correct predictions.*[6]

Let (d^n, c^n) be the configuration of $\mathcal{C}_{\mathcal{A}}$ encoding the initial configuration of $\mathcal{B}^n$, *i.e.*, such that $(d^n, c^n) \approx H(\ell_0, u_n)$. Lemmas 4, 6 and 7 then yield:

Theorem 8. *Let $n \geq 3$. $\mathcal{C}_{\mathcal{A}}$ has a time-divergent*[7] *computation starting in (d^n, c^n) iff $\mathcal{L}([\![\mathcal{B}^n]\!]) \neq \varnothing$.*

The second part of the construction of the CAROT for encoding the robust model-checking problem consists in adding the part related to the temporal formula φ (in MTL). This part will be handled in a very similar way as in [7], we thus simply sketch the construction. First, we build the one-clock alternating timed automaton $\mathcal{A}_{\neg\varphi}$ corresponding to $\neg\varphi$. Then, we build the product of

[6] Intuitively, this is because delay transitions force predictions checking.
[7] That is with infinitely many delay transitions.

the CAROT $\mathcal{C}_{\mathcal{A}}$ with a CAROT simulating the behaviour of $\mathcal{A}_{\neg\varphi}$. The resulting CAROT, say $\mathcal{C}_{\mathcal{A},\neg\varphi}$, running from the initial configuration $(d^{\varphi,n}, c^{\varphi,n})$ corresponding via $\approx$ to the initial configuration of $\mathcal{B}^n \times \mathcal{A}_{\neg\varphi}$, simulates joint behaviours of $\mathcal{B}^n$ and $\mathcal{A}_{\neg\varphi}$. The accepting condition for $\mathcal{C}_{\mathcal{A},\neg\varphi}$ is the Büchi condition given by 'flattening' $\mathcal{A}_{\neg\varphi}$. The results of [7] combined with our above results yield:

Theorem 9. *Let $n \geq 3$ and $\varphi \in \mathsf{MTL}$. $\mathcal{C}_{\mathcal{A},\neg\varphi}$ has a time-divergent accepting computation starting in $(d^{\varphi,n}, c^{\varphi,n})$ iff $[\![\mathcal{B}^n]\!] \not\models \varphi$.*

Remark. A rough bound on the size of the CAROT $\mathcal{C}_{\mathcal{A},\neg\varphi}$ is $O\Big(|\mathcal{A}|^3 \cdot 2^{O(M\cdot|\varphi|+|X|)}\Big)$, where $|\varphi|$ is the number of subformulae of φ. The size of the alphabet of $\mathcal{C}_{\mathcal{A},\neg\varphi}$ is in $O(|X|)$. As proved in [7], if $\mathcal{C}_{\mathcal{A},\neg\varphi}$ has an h-cycle-bounded accepting execution, then there is a bound N_0, which depends on the size of $\mathcal{C}_{\mathcal{A},\neg\varphi}$ and h, such that there exists an h-cycle-bounded accepting execution of length no more than N_0. We do not give the precise value of this bound (see [10] instead), but it is exponential in h, which is itself exponential in the size of the input.

3.3 From CAROTs to Robust Model Checking

We first solve the robust model-checking problem for $\mathsf{Bounded\text{-}MTL}$, and then turn to the more involved logic $\mathsf{coFlat\text{-}MTL}$. Both rely on the previously proved equivalences:

$$\mathcal{A} \not\models \varphi \quad \text{iff} \quad \forall n \geq 3,\ \begin{cases} \mathcal{C}_{\mathcal{A},\neg\varphi} \text{ has a time-divergent accepting} \\ \text{computation starting in } (d^{\varphi,n}, c^{\varphi,n}) \end{cases}$$

Robust model-checking for Bounded-MTL. The algorithm to decide the robust model-checking problem for $\mathsf{Bounded\text{-}MTL}$ formula relies on the fact that the truth value of a $\mathsf{Bounded\text{-}MTL}$ formula φ along a run ϱ only depends on the first h time units of ϱ, where h is the sum of the constants appearing in φ [7]. In $\mathcal{A}_{\neg\varphi}$, after having read a prefix of duration h time units, we thus always end up in a sink state, that we report as accepting in the CAROT.

Moreover, the non-blocking assumption made on $\mathcal{A}$ implies the following property:

Lemma 10. *Let $\mathcal{A}$ be a timed automaton, and $\delta > 0$. Given (ℓ, v) a configuration of $\mathcal{A}$ such that $v \models_\delta I(\ell)$, we have that $\mathcal{L}([\![\mathcal{A}]\!]_\delta^{(\ell,v)}) \neq \varnothing$.*

Hence, robustly model-checking $\mathcal{A}$ against a $\mathsf{Bounded\text{-}MTL}$ property φ will be reduced to searching, for every $n \geq 3$, for a time-bounded accepting prefix in $\mathcal{T}^n_{\mathcal{C}_{\mathcal{A},\neg\varphi}}$, and verifying that the reachable configuration is with correct predictions. Indeed, applying the previous lemma, we already know that we will be able to extend this finite prefix into a time-diverging run witnessing $\neg\varphi$ as soon as the prefix is correctly chosen (meaning it ends up in a accepting state of the CAROT).

We define the following property, for any integer n:

$$\mathcal{P}(n)\text{: “}\mathcal{C}_{\mathcal{A},\neg\varphi} \text{ has an } h\text{-cycle-bounded accepting computation with correct predictions starting in } (d^{\varphi,n}, c^{\varphi,n})\text{”}$$

Then our problem somehow amounts to checking that for every $n \geq 3$, property $\mathcal{P}(n)$ holds. This is some kind of "universality" checking of the CAROT, where we universally quantify on the initial number of Δ's on the channel. This is achieved using the following two lemmas:

Lemma 11. *Let $n, n' \in \mathbb{N}$ be such that $n' \geq 2n \geq 6$. Then $\mathcal{P}(n') \Rightarrow \mathcal{P}(n)$.*

Proof. We have seen that

$$[\![\mathcal{B}^{n'}]\!] \underset{\text{(Lemma 2)}}{\sqsupseteq} [\![\mathcal{A}_{\frac{2}{n'}}]\!] \underset{(\frac{2}{n'} \leq \frac{1}{n})}{\sqsupseteq} [\![\mathcal{A}_{\frac{1}{n}}]\!] \underset{\text{(Lemma 2)}}{\sqsupseteq} [\![\mathcal{B}^{n}]\!].$$

Also, for $m \geq 3$, the CAROT $\mathcal{C}_{\mathcal{A},\neg\varphi}$, when restricted to configurations with correct predictions, is time-abstract bisimilar to the product of $\mathcal{B}^m$ with $\mathcal{A}_{\neg\varphi}$. Finally, the respective initial configurations are in the relation $\approx$. □

Lemma 12. *Let $N \geq 2 \cdot N_0$. Then $\mathcal{P}(N) \Rightarrow \forall n \in \mathbb{N}, \exists n' \geq n$ s.t. $\mathcal{P}(n')$.*

Proof (Sketch). Using the notion of computation table introduced in [7], we prove a pumping lemma for CAROTs. Indeed, the height of the computation table is bounded by h; the number of "sliding windows" of such tables is thus bounded. Hence, once the table is large enough, it is possible to duplicate one of its fragments, building new computation tables encoding computations over larger inputs (corresponding to configurations $(d^{\varphi,n'}, c^{\varphi,n'})$ for integers n' arbitrarily larger than n). □

Thanks to those lemmas, it suffices to only look for an h-cycle-bounded execution starting in one of the configurations $(d^{\varphi,N}, c^{\varphi,N})$ (for any $N \geq 2 \cdot N_0$) in order to ensure the existence of an accepting execution for any number of Δ's:

Corollary 13. *Let $N \geq 2 \cdot N_0$. Then $\mathcal{P}(N) \Leftrightarrow \forall n \geq 3,\ \mathcal{P}(n)$.*

Theorem 14. *The model-checking problem for* Bounded-MTL *is* EXPSPACE-*Complete (and* PSPACE-*Complete if constants of the formula are given in unary).*

Proof. The hardness parts follow from the same hardness results for Bounded-MTL satisfiability [7].

The upper bound follows from the previous study. However, since the size of the CAROT is doubly exponential, the non-deterministic algorithm of Theorem 1 has to be applied on-the-fly, without explicitly building the CAROT. Since the number N_0 of different sliding windows of height h is also doubly-exponential in the size of the input, our non-deterministic algorithm will also have a counter, and will stop as soon as the counter reaches N_0. This all can be achieved within exponential space.

If the constants of the formula are unary-encoded, then h is linear in the size of the input formula, and N_0 is simply exponential in the size of the input. The same algorithm then uses only polynomial space. □

Robust model-checking for coFlat-MTL. The case of coFlat-MTL is more involved than that of Bounded-MTL. The reason is that, unlike Bounded-MTL, the truth value of a coFlat-MTL formula φ along a run ϱ does not only depend on a prefix of ϱ of bounded duration. Instead, we have the following decomposition lemma, which follows from [7, Theorem 12]:

Lemma 15. *Let ϖ be an accepting run of $[\![\mathcal{A}]\!]_\delta \times [\![\mathcal{A}_{\neg\varphi}]\!]$. Then it can be decomposed as $\varpi_1 \cdot \varpi_2 \cdot \varpi_3 \cdots \varpi_{2m}$ where there is a finite automaton $\mathcal{F}_{\neg\varphi}$ [8] s.t.:*

(i) the duration of ϖ_{2i-1} (for $1 \leq i \leq m$) is bounded by $h = (2M + 3 + W \cdot 2^{|\varphi|}) \cdot (|\varphi| \cdot 2^{|\varphi|})$,
(ii) the duration of ϖ_{2i} (for $1 \leq i \leq m$) is at least $2^{|\varphi|} \cdot (2W + 1)$, and along that portion, the behaviour of $\mathcal{A}_{\neg\varphi}$ is that of $\mathcal{F}_{\neg\varphi}$,
(iii) the Büchi condition of $\mathcal{F}_{\neg\varphi}$ is satisfied along ϖ_{2m}, and
(iv) $2m \leq |\varphi| \cdot 2^{|\varphi|}$,

where W is the number of states of the region automaton of $\mathcal{A} \times \mathcal{F}_{\neg\varphi}$. Odd-numbered segments are called the active parts*, while even-numbered ones are said* inactive.

This decomposition lemma inspires a decidability algorithm, where we modularly check the existence of runs not satisfying φ by distinguishing between active and inactive parts of the runs. Indeed, given a sequence $(\varrho^\delta)_{\delta>0}$, using combinatorics arguments, it is possible to twist them so that, for $\delta > 0$ small enough (say $\delta \leq \delta_0$), all ϱ^δ look very similar (that is, roughly the junction points between active and inactive parts are close to each other and belong to the same region). Conversely, if we are given a witnessing run ϱ^{δ_0}, and if we consider the junction points of that run, for each inactive (resp. active) part, it is possible for every $\delta > 0$ to build an inactive (resp. active) portion of a run joining the two junction points. The construction of an inactive portion of a run partly relies on approximation results proved in [15, 8], and the construction of an active portion of a run relies on a result similar to Corollary 13. The complete proof is rather technical and gathers in an original way many results of [15, 8, 7].

An informal version of the decidability algorithm is the following, and can be schematised as on Figure 3:

- Guess the junction points of the active and inactive parts
- For each active part, check that the two guessed junction points are reachable in $\mathcal{C}_{\mathcal{A},\neg\varphi}$ in a cycle-bounded manner[9] (here, we need to prove a result similar to Corollary 13)
- For each inactive and bounded active part, check that the two junction points are "robustly" reachable (using results of [15, 8])
- For the last unbounded active part, check that the automaton $\mathcal{A} \times \mathcal{F}_{\neg\varphi}$, from last junction point, does not robustly satisfy the acceptance condition of $\mathcal{F}_{\neg\varphi}$ interpreted as a co-Büchi condition (using the algorithm of [9]).

We can conclude with the main result of the paper:

[8] Intuitively, $\mathcal{F}_{\neg\varphi}$ is obtained as the flattening of the untimed part of $\mathcal{A}_{\neg\varphi}$, see [10].
[9] The bound on the number of cycles is that of (i) in the above decomposition lemma.

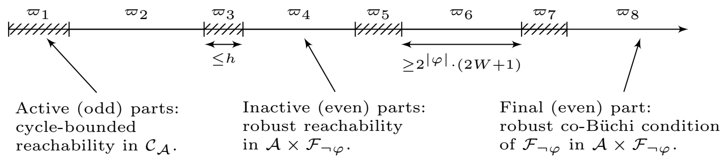

Fig. 3. Global view of our algorithm

Theorem 16. *The robust model-checking problem for* coFlat-MTL *is* EXPSPACE-*Complete.*

4 Conclusion

In this paper, we have proposed a new approach to robust model-checking of timed systems based on channel machines: we construct a family of networks of timed systems such that robustly verifying a formula in a timed automaton reduces to the verification of the formula in one of the members of the family; Then we encode the behaviour of this family of timed systems using channel machines. We have applied this approach to coFlat-MTL, a rather expressive fragment of MTL, and prove that it can be decided in EXPSPACE, which is moreover optimal. The logic coFlat-MTL subsumes LTL, thus it is the more general specification language for which robust model-checking has been proved decidable.

Our correctness proofs heavily rely on technical lemmas proved in [15, 8] and is unfortunately not fully CAROT-based. As future works, we plan to study robust reachability directly on the CAROT encoding the extended semantics, in order to develop a fully CAROT-based algorithm for coFlat-MTL.

References

1. K. Altisen and S. Tripakis. Implementation of timed automata: An issue of semantics or modeling? In *Proc. 3rd Intl Conf. Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, LNCS 3829, pages 273–288. Springer, 2005.
2. R. Alur and D. Dill. A theory of timed automata. *Theor. Comp. Sci.*, 126(2):183–235, 1994.
3. R. Alur, T. Feder, and Th. A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996.
4. R. Alur, S. La Torre, and P. Madhusudan. Perturbed timed automata. In *Proc. 8th Intl Workshop Hybrid Systems: Computation & Control (HSCC'05)*, LNCS 3414, pages 70–85. Springer, 2005.
5. C. Baier, N. Bertrand, P. Bouyer, Th. Brihaye, and M. Größer. Almost-sure model checking of infinite paths in one-clock timed automata. Research Report LSV-07-29, ENS Cachan, France, 2007.
6. C. Baier, N. Bertrand, P. Bouyer, Th. Brihaye, and M. Größer. Probabilistic and topological semantics for timed automata. In *Proc. 27th Conf. Found. Softw. Tech. & Theor. Comp. Sci. (FSTTCS'07)*, LNCS 4855, pages 179–191. Springer, 2007.

7. P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. The cost of punctuality. In *Proc. 22nd Ann. Symp. Logic in Computer Science (LICS'07)*. IEEE Comp. Soc. Press, 2007. 109–118.
8. P. Bouyer, N. Markey, and P.-A. Reynier. Robust model-checking of timed automata. Research Report LSV-05-06, ENS Cachan, France, 2005.
9. P. Bouyer, N. Markey, and P.-A. Reynier. Robust model-checking of timed automata. In *Proc. 7th Latin American Symp. Theoretical Informatics (LATIN'06)*, LNCS 3887, pages 238–249. Springer, 2006.
10. P. Bouyer, N. Markey, and P.-A. Reynier. Robust analysis of timed automata via channel machines. Research Report LSV-07-32, ENS Cachan, France, 2007.
11. M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos: a model-checking tool for real-time systems. In *Proc. 10th Intl Conf. Computer Aided Verification (CAV'98)*, LNCS 1427, pages 546–550. Springer, 1998.
12. F. Cassez, Th. A. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In *Proc. 5th Intl Workshop Hybrid Systems: Computation & Control (HSCC'02)*, LNCS 2289, pages 134–148. Springer, 2002.
13. C. Daws and P. Kordy. Symbolic robustness analysis of timed automata. In *Proc. 4th Intl Conf. Formal Modeling and Analysis of Timed Systems (FORMATS'06)*, LNCS 4202, pages 143–155. Springer, 2006.
14. M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robustness and implementability of timed automata. In *Proc. Joint Conf. Formal Modelling and Analysis of Timed Systems & Formal Techniques in Real-Time and Fault Tolerant System (FORMATS+FTRTFT'04)*, LNCS 3253, pages 118–133. Springer, 2004.
15. M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robustness and implementability of timed automata. Tech. Report 2004.30, Centre Fédéré en Vérification, Belgium, Dec. 2005. Revised version.
16. M. De Wulf, L. Doyen, and J. Raskin. Almost ASAP semantics: From timed models to timed implementations. In *Proc. 7th Intl Workshop Hybrid Systems: Computation & Control (HSCC'04)*, LNCS 2993, pages 296–310. Springer, 2004.
17. C. Dima. Dynamical properties of timed automata revisited. In *Proc. 5th Intl Conf, Formal Modeling and Analysis of Timed Systems (FORMATS'07)*, LNCS 4763, pages 130–146. Springer, 2007.
18. T. French, J. C. McCabe-Dansted, and M. Reynolds. A temporal logic of robustness. In *Proc. 6th Intl Workshop Frontiers of Combining Systems (FroCoS'07)*, LNAI 4720, pages 193–205. Springer, 2007.
19. V. Gupta, Th. A. Henzinger, and R. Jagadeesan. Robust timed automata. In *Proc. Intl Workshop Hybrid and Real-Time Systems (HART'97)*, LNCS 1201, pages 331–345. Springer, 1997.
20. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.
21. K. G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *J. Software Tools for Technology Transfer*, 1(1–2):134–152, 1997.
22. J. Ouaknine and J. Worrell. Revisiting digitization, robustness and decidability for timed automata. In *Proc. 18th Ann. Symp. Logic in Computer Science (LICS'03)*. IEEE Comp. Soc. Press, 2003.
23. J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In *Proc. 19th Ann. Symp. Logic in Computer Science (LICS'05)*, pages 188–197. IEEE Comp. Soc. Press, 2005.
24. J. Ouaknine and J. Worrell. On the decidability and complexity of Metric Temporal Logic over finite words. *Logical Methods in Comp. Sci.*, 3(1-8):1–27, 2007.

25. A. Pnueli. The temporal logic of programs. In *Proc. 18th Ann. Symp. Foundations of Computer Science (FOCS'77)*, pages 46–57. IEEE Comp. Soc. Press, 1977.
26. A. Puri. Dynamical properties of timed automata. In *Proc. 5th Intl Symp. Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'98)*, LNCS 1486, pages 210–227. Springer, 1998.
27. M. Swaminathan and M. Fränzle. A symbolic decision procedure for robust safety of timed systems. In *Proc. 14th Intl Symp. Temporal Representation and Reasoning (TIME'07)*, page 192. IEEE Comp. Soc. Press, 2007.